

DWES

Acceso a datos (PDO). Consultas
2º DAW

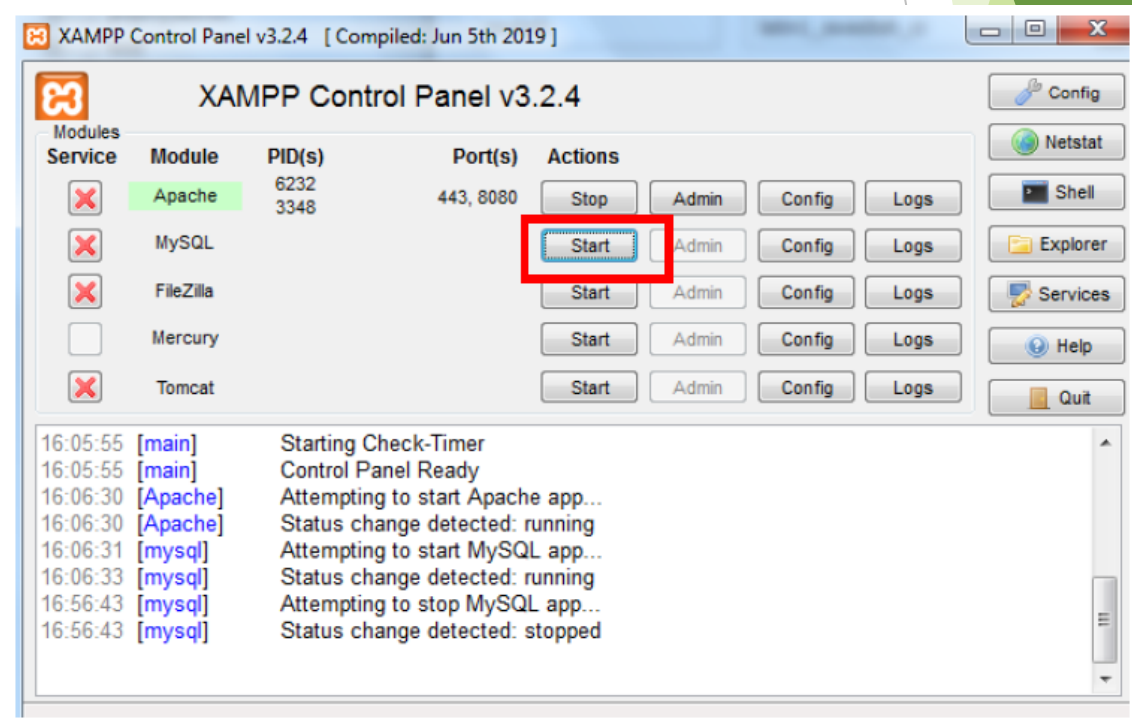
Trabajar con BBDD en PHP

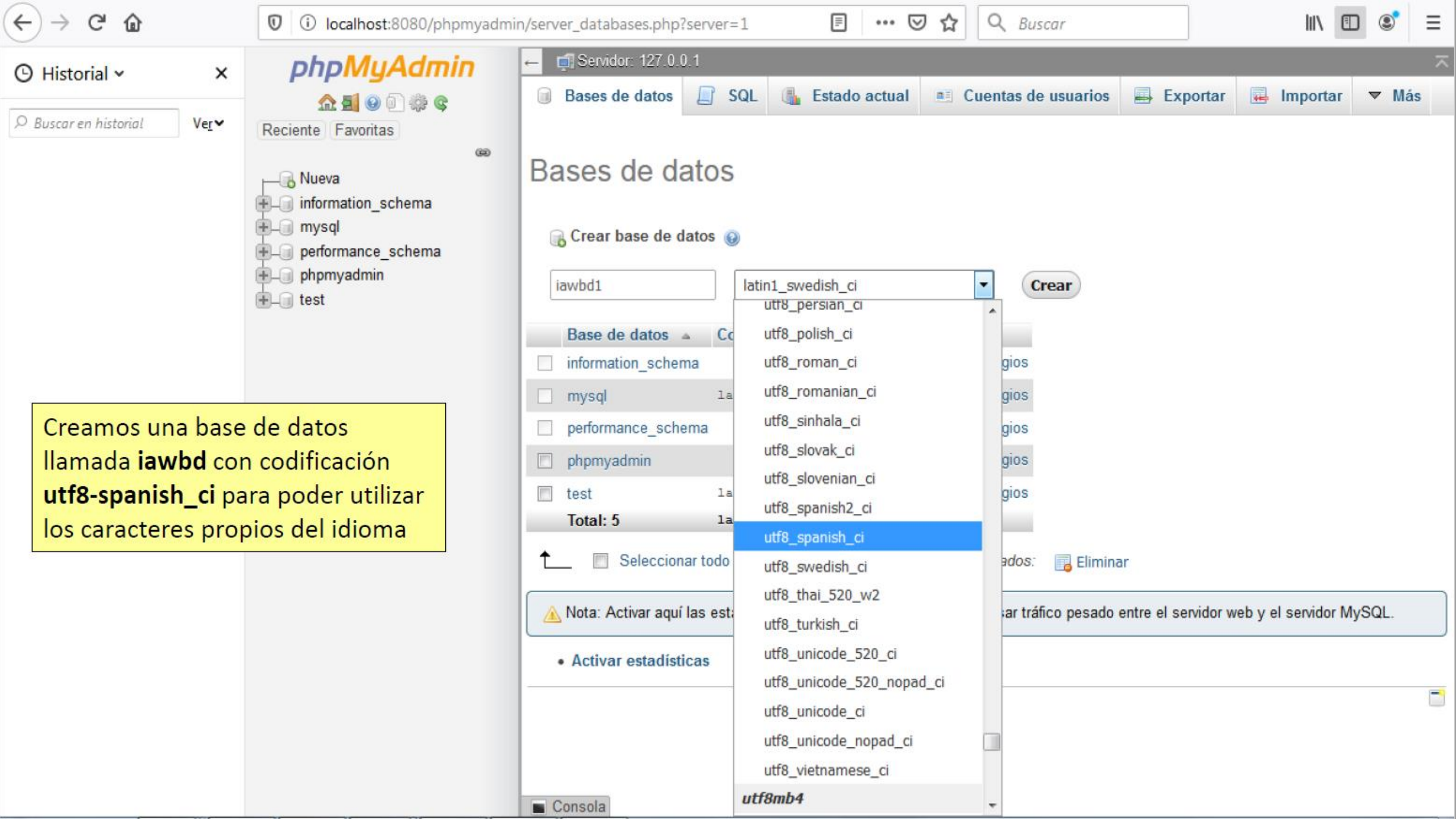
Antes de empezar a trabajar con BBDD en PHP vamos a aprender a manejar la herramienta **PHPMyAdmin** que viene incorporada en XAMPP.

Vamos a crear desde la herramienta una base de datos y una tabla. Más adelante:

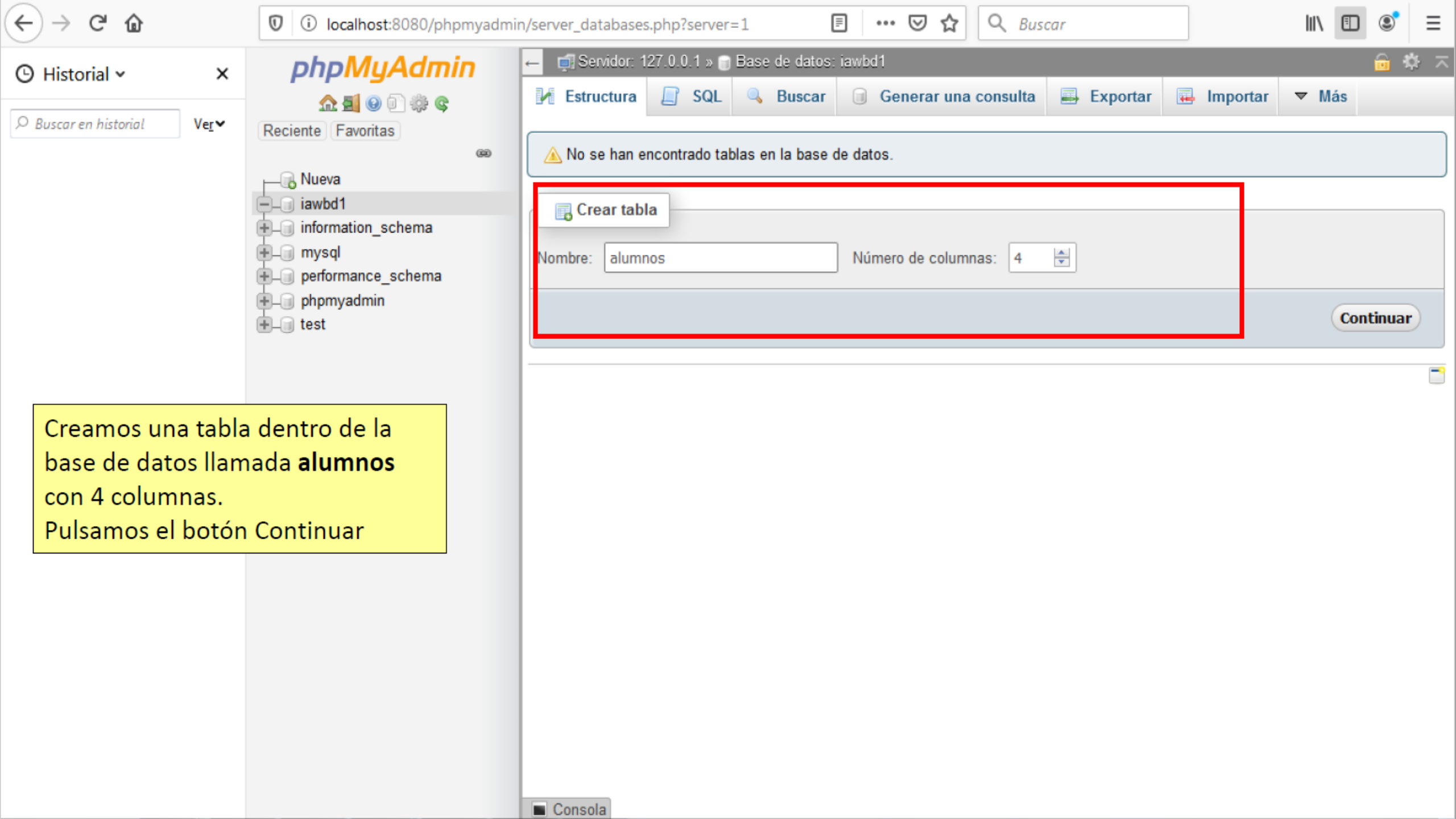
- Importaremos una tabla.
- Exportaremos una base de datos completa.
- Borraremos una tabla y una base de datos.
- Importaremos una base de datos

Antes de empezar, recuerda tener **inicializado en el panel de control de xampp el gestor de bases de datos.** Con MySQL iniciado podemos pulsar en Admin para empezar a gestionar la herramienta.





Creamos una base de datos llamada **iawbd** con codificación **utf8-spanish_ci** para poder utilizar los caracteres propios del idioma



Creamos una tabla dentro de la base de datos llamada **alumnos** con 4 columnas.
Pulsamos el botón Continuar

Trabajar con BBDD en PHP

❖ **Al crear la tabla**, se almacenarán datos de alumnos que desarrollarán cursos de programación en PHP, ASP y JSP. Tendrá los siguientes campos:

- Código del alumno, de tipo numérico (int), será la clave primaria y al indicar que es auto_incremental generará automáticamente por el gestor de base de datos.
- Los campos nombre y mail son de tipo varchar(podemos almacenar cualquier carácter)
- Código Curso representa el curso a tomar por el alumno (1=PHP, 2=ASP y 3=JSP)

Nombre de la tabla: Agregar columna(s)

Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento	Atributo
<input type="text" value="codigo"/> <small>Seleccionar desde las columnas centrales</small>	INT		Ninguno		<input type="checkbox"/> PRIMARY <input checked="" type="checkbox"/>
<input type="text" value="nombre"/> <small>Seleccionar desde las columnas centrales</small>	VARCHAR	50	Ninguno		
<input type="text" value="mail"/> <small>Seleccionar desde las columnas centrales</small>	VARCHAR	70	Ninguno		
<input type="text" value="codigocurso"/> <small>Seleccionar desde las columnas centrales</small>	INT		Ninguno		

Para crear la tabla pulsamos el botón Guardar

Trabajar con BBDD en PHP

❖ Ahora vamos a **borrar la tabla alumnos**. Para ello selecciona la tabla de la base de datos en el panel izquierdo de la pantalla. Puedes borrar la tabla de dos formas:

- 1) Seleccionando la opción del menú superior “Más -> Operaciones” y en la pantalla que aparece seleccionar la opción remarcada en rojo “Borrar la tabla (DROP)”.
- 2) Seleccionando a opción del menú superior “SQL” y escribir la sentencia SQL que borra la tabla. **“DROP TABLE ALUMNOS;”** y pulsar el botón continuar.

❖ Para **borrar la base de datos completa** debes tener seleccionada la base de datos pinchando con el ratón sobre su nombre (hay que seleccionar la BD no una tabla). A continuación, podemos:

- 1) Ejecutar la sentencia **“DROP DATABASE IAWDB;”** pulsando el menú SQL y escribiendo la sentencia, igual que en el ejemplo anterior.
- 2) Seleccionar la opción del menú superior “Operaciones” y en la pantalla que aparece seleccionar la opción que aparece remarcada en color rojo **“Eliminar la base de datos (DROP)”** y pulsar el botón “Continuar”



Trabajar con BBDD en PHP

- Es habitual necesitemos mostrar en una página Web información extraída de una BD, o realizar **consultas de actualización, borrado o inserción de datos**.
- Tradicionalmente las conexiones se establecían utilizando la extensión nativa MySQL. Esta extensión se mantiene en la actualidad para dar soporte a las aplicaciones ya existentes que la utilizan, pero no se recomienda utilizarla para desarrollar nuevos programas.
- Lo más habitual es elegir entre:
 - ✓ **MySQLi (extensión nativa)**
 - ✓ **PDO.**
- La mayor diferencia entre MySQLi y PDO es que **PDO es independiente de la BD que utilizemos**. En PDO si tenemos que cambiar un proyecto para usar otra base de datos, solo habría que cambiar la cadena de conexión y algunas consultas, mientras que con MySQLi, hay que volver a escribir todo el código, incluidas las consultas.

Trabajar con BBDD en PHP

- Si en el futuro existe la posibilidad de utilizar otro servidor como almacenamiento y que haya que cambiar el SGBD por otro distinto la opción más recomendable en la actualidad es usar la extensión PDO.
- A partir de la versión 5 de PHP se introdujo la extensión PDO (PHP DATA OBJECTS) para acceder de una forma común a distintos sistemas gestores de BBDD.
- La extensión PDO debe utilizar un driver o controlador específico para el tipo de base de datos que se utilice. Para consultar los controladores disponibles en tu instalación de PHP, puedes utilizar la información que proporciona la función `phpinfo`.
- PDO se basa en las características de orientación a objetos de PHP. Para acceder a las funcionalidades de la extensión tienes que emplear los objetos que ofrece, con sus métodos y propiedades.

Trabajar con BBDD en PHP

❖ PDO: establecer conexiones

Para establecer una conexión con una base de datos utilizando PDO, hay que instanciar (crear) un objeto de la clase PDO pasándole los siguientes parámetros (solo el primero es obligatorio):

- ✓ **Origen de datos (DSN).** Es una cadena de texto que indica qué controlador se va a utilizar (nosotros usaremos el controlador de mysqli) y a continuación, separadas por el carácter dos puntos, los parámetros específicos necesarios por el controlador, como por ejemplo el nombre o dirección IP del servidor y el nombre de la base de datos.
- ✓ **Nombre de usuario** con permisos para establecer la conexión.
- ✓ **Contraseña** del usuario.
- ✓ **Opciones de conexión**, almacenadas en forma de array.

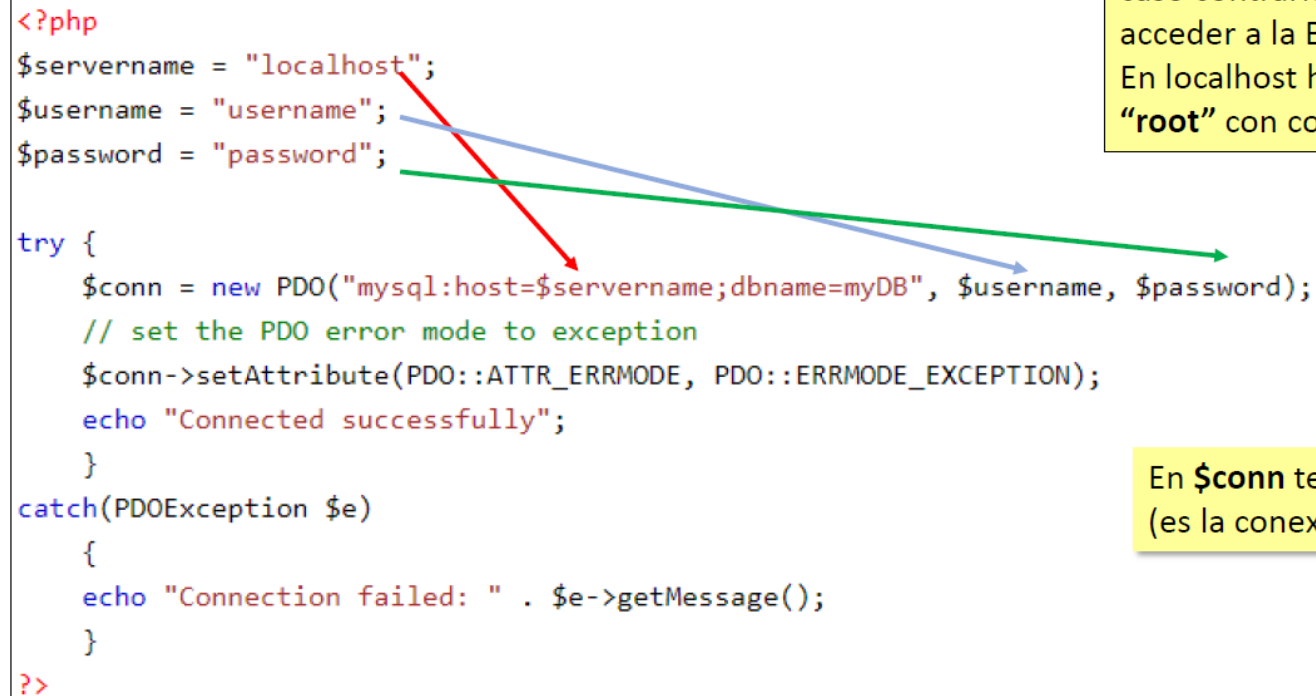
Trabajar con BBDD en PHP

❖ PDO: establecer conexiones

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
}
catch(PDOException $e)
{
    echo "Connection failed: " . $e->getMessage();
}

?>
```



El usuario y la contraseña deben estar definidos una cuenta de usuario, en caso contrario dará error al intentar acceder a la BD.
En localhost hay definido el usuario "root" con contraseña ""

En **\$conn** tenemos un objeto PDO creado (es la conexión a la BD).

Trabajar con BBDD en PHP

❖ PDO: establecer conexiones

Si utilizamos la extensión PDO podemos indicaremos el conjunto de caracteres con el que queremos trabajar directamente en la cadena de conexión:

```
$conn = new PDO("mysql:host=$servidor;dbname=myDBPDO;charset=utf8",  
                $usuario, $clave);
```

La extensión PDO permite definir qué queremos que ocurra cuando se produzca un error, utilizando para ello el atributo PDO::ATTR_ERRMODE. Las posibilidades son:

- **PDO::ERRMODE_SILENT.** No se hace nada cuando ocurre un error. Es el comportamiento por defecto.
- **PDO::ERRMODE_WARNING.** Genera un error de tipo E_WARNING cuando se produce un error.
- **PDO::ERRMODE_EXCEPTION.** Cuando se produce un error lanza una excepción utilizando el manejador propio PDOException. Este es el que utilizaremos.

Trabajar con BBDD en PHP

❖ PDO: consultas

Para ejecutar una consulta SQL utilizando PDO, debes diferenciar aquellas sentencias SQL que no devuelven como resultado un conjunto de datos, de aquellas que sí lo devuelven.

- Para realizar consultas que no devuelven resultado (**INSERT, DELETE o UPDATE**), se usa el **método exec()** que devuelve el número de registros afectados.
- Para realizar consultas que devuelven un conjunto de datos (**SELECT**) se utiliza el **método query()** que devuelve un objeto de la clase **PDOStatement**

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");  
$resultado = $dwes->query("SELECT producto, unidades FROM stock");
```

```
$registros = $dwes->exec('DELETE FROM stock WHERE unidades=0');  
print "<p>Se han borrado $registros registros.</p>";
```

Trabajar con BBDD en PHP

❖ PDO: crear una BBDD

```
<?php
$servidor= "localhost";
$usuario= "root";
$clave= "";
$bd= "myDBPDO";
try {
    $conn = new PDO("mysql:host=$servidor;dbname=$bd;charset=utf8", $usuario, $clave);

    // ponemos el modo error PDO (EXCEPCIONES)
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    //AQUÍ VA EL CÓDIGO QUE OPERA CON LA BD
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}
$conn = null;
?>
```

Éste es el código básico con el que vamos a trabajar. Realiza la conexión, establece el modo excepción y si se produce alguna muestra el mensaje de la excepción que ha saltado, y cerramos la conexión.

- El código que queremos ejecutar lo escribimos entre las llaves de la palabra reservada try.
- El bloque catch se encarga de recoger la excepción (si ocurre) y en este caso muestra en pantalla el error con el método getMessage(), que devuelve el mensaje de la excepción ocurrida.
- Para que salten las excepciones añadimos el atributo ATTR_ERRMODE (modo error)

Trabajar con BBDD en PHP

❖ PDO: crear una BBDD

```
$servidor= "localhost";
$usuario= "root";
$clave= "";
try {
    $conn = new PDO("mysql:host=$servidor", $usuario, $clave);
    // ponemos el modo error PDO (EXCEPCIONES)
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "CREATE DATABASE myBDPDO "
        . "DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_general_ci;";
    // usamos el método exec() porque no se devuelven resultados.
    $conn->exec($sql);
    echo "Base de datos myBDPDO creada correctamente<br>";
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
```

Observa que cuando vamos a crear una base de datos, en la instrucción de la conexión, **no parece el nombre de la bd**, (no existe, la vamos a crear)

- Guardamos en una variable el texto de la sentencia SQL a ejecutar.
- Como es una sentencia que no devuelve un conjunto de resultados se ejecuta mediante el método `exec` del objeto que tiene la conexión pasando la sentencia a ejecutar como parámetro.
- Para cerrar la conexión asignamos el valor `null` al puntero.
- Fíjate en que la forma de llamar a un método es utilizando el objeto que tiene la conexión (`$conn`), seguido de `->` y a continuación el nombre del método.
- Si la BD ya existe saltará una excepción.

Trabajar con BBDD en PHP

❖ PDO: crear una tabla en una BBDD

Similar al ejemplo anterior.

Cambia el texto de la sentencia SQL a ejecutar.

Si la tabla ya existe saltará una excepción.

Copia el código y ejecútalo.

Comprueba en PHPMyAdmin que ha creado la tabla.

```
$bd= "myDBPDO";
try {
    $conn = new PDO("mysql:host=$servidor;dbname=$bd", $usuario, $clave);
    // ponemos el modo error de PDO a excepción
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "CREATE TABLE ALUMNOS ( CODIGO INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,"
        . " NOMBRE VARCHAR(30) NOT NULL, APELLIDOS VARCHAR(30) NOT NULL, TELEFONO INT(9), "
        . "CORREO VARCHAR(50) ) DEFAULT CHARACTER SET utf8
        DEFAULT COLLATE utf8_general_ci;";

    //se utiliza el método exec() porque la sentencia SQL no devuelve ningún resultado.
    $conn->exec($sql);
    echo "Se creó la tabla alumnos en myDBPDO correctamente <br/>";
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}

?>
```


Trabajar con BBDD en PHP

❖ PDO: conjunto de resultados

Cuando ejecutamos una sentencia SELECT usando el método query() obtenemos un conjunto de resultados. En PDO tenemos varias posibilidades para tratar con el conjunto de resultados devuelto por el método query().

La más utilizada es el **método fetch()** que devuelve un registro del conjunto de resultados, o false si ya no quedan registros por recorrer.

Para tratar todos los registros devueltos, utilizaremos un bucle. En cada vuelta del bucle recuperaremos un nuevo registro hasta que no haya más resultados.

```
$resultado=$conn->query("SELECT * FROM alumnos");  
while ($row = $resultado->fetch(PDO::FETCH_ASSOC)) {  
    ...  
}
```


Trabajar con BBDD en PHP

❖ PDO: conjunto de resultados

```
$resultado=$conn->query("SELECT * FROM alumnos");  
while ($row = $resultado->fetch(PDO::FETCH_ASSOC)) {  
    ;  
}
```

Por defecto, el método fetch genera y devuelve un array con claves numéricas y asociativas (la clave asociativa corresponde con el nombre de la columna definida en la tabla de la BD) de un registro.

Para cambiar su comportamiento, admite un parámetro opcional que puede tomar (entre otros) uno de los siguientes valores:

- **PDO::FETCH_ASSOC.** Devuelve solo un array asociativo.
- **PDO::FETCH_NUM.** Devuelve solo un array con claves numéricas.

Trabajar con BBDD en PHP

❖ PDO: conjunto de resultados

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");  
$resultado = $dwes->query("SELECT producto, unidades FROM stock");  
while ($registro = $resultado->fetch()) {  
    echo "Producto ".$registro['producto'].": ".$registro['unidades']."  
}  
}
```

Se recorre el conjunto de resultados devuelto mediante un bucle while.

Guardamos en la variable \$registro cada fila del conjunto de resultados devuelto con fetch() (corresponde a un registro de la tabla de la BD).

En cada iteración del bucle, la variable \$registro contiene un array con los campos de cada fila de la tabla. Las claves del array para acceder a cada campo corresponden con el nombre que tiene cada columna en la tabla de la base de datos.

En la primera vuelta del bucle, \$registro['producto'] contiene "tele", y \$registro['unidades'] contiene 50.

El bucle finaliza cuando no hay más registros.

producto	unidades
tele	50
cámara	200
PC	20
portátil	500
móvil	150
tablet	30
...	...
smartwatch	300

Trabajar con BBDD en PHP

❖ PDO: conjunto de resultados

```
<?php
    $servidor = "localhost";
    $usuario = "root";
    $clave = "";
    $sql="";

    $sql= "SELECT * FROM ALUMNOS";

    try {
        $conn = new PDO("mysql:host=$servidor;dbname=myDBPDO;charset=utf8", $usuario, $clave);
        // asignamos el modo excepción
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        echo "<h2>listado de la tabla alumnos </h2>";
        echo "<table>";
        echo "<tr><th>CÓDIGO</th><th>NOMBRE</th><th>APELLIDOS</th><th>TELÉFONO</th><th>CORREO</th></tr>";
        foreach ($conn->query($sql) as $row) {
            echo "<tr><td>".$row["CODIGO"]. "</td><td>".$row["NOMBRE"]. "</td><td>"
                . $row["APELLIDOS"]. "</td><td>".$row["TELEFONO"]. "</td><td>".$row["CORREO"]
                . "</td></tr>";
        }
        echo "</table>";
        echo "<br/><hr/><br/>";
    }
    catch(PDOException $e)
    {
        // deshace la transacción si algo ha fallado.
        echo "Error: " . $e->getMessage();|
    }
    $conn = null;
?>
```

Creamos dinámicamente una tabla HTML para mostrar los resultados de la consulta.

En la bucle while definimos que los resultados nos lo devuelve únicamente como un array asociativo. La clave de cada elemento será el nombre de la columna definido en la tabla de la BD.

Trabajar con BBDD en PHP

❖ PDO: consultas preparadas

- Utilizando PDO podemos preparar en el servidor consultas con parámetros para ejecutarlas de forma repetida.
- Las declaraciones preparadas protegen de la inyección de SQL y son muy importantes para la seguridad de las aplicaciones web.
- Para preparar la consulta en el servidor MySQL, se debe utilizar el **método prepare()** de la clase PDO que **devuelve un objeto de la clase PDOStatement**.
- A continuación, se asignan parámetros a las variables de la sentencia.
- Por último, se ejecuta la sentencia preparada, se utiliza el **método execute()**.

Trabajar con BBDD en PHP

❖ PDO: consultas preparadas

Los parámetros se pueden marcar utilizando signos de interrogación o utilizando parámetros con nombre, precediéndolos por el símbolo de dos puntos.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");  
$consulta = $dwes->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');
```

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");  
$consulta = $dwes->prepare('INSERT INTO familia (cod, nombre) VALUES (:cod, :nombre)');
```

Trabajar con BBDD en PHP

❖ PDO: consultas preparadas

Antes de ejecutar la consulta hay que asignar un valor a los parámetros utilizando el método **bindParam()** de la clase PDOStatement.

- Si utilizas signos de interrogación para marcar los parámetros, debes indicar el número de columna de la tabla de la base de datos al que corresponde ese parámetro, y a continuación la variable con el valor a asignar.
- Si utilizas parámetros con nombre, debes indicar ese nombre en el primer parámetro del método `bindParam()` , y a continuación la variable con valor a asignar.

```
$cod_producto = "TABLET";  
$nombre_producto = "Tablet PC";  
$consulta->bindParam(1, $cod_producto);  
$consulta->bindParam(2, $nombre_producto);
```

Trabajar con BBDD en PHP

❖ PDO: consultas preparadas

```
$consulta->bindParam(":cod", $cod_producto);  
$consulta->bindParam(":nombre", $nombre_producto);
```

Siempre hay que usar variables para asignar los parámetros en una consulta preparada, no se pueden utilizar literales, si no se producirá un error.

Una vez preparada la consulta y enlazados los parámetros con sus valores, se ejecuta la consulta utilizando el método **execute**.

```
$consulta->execute();
```

Trabajar con BBDD en PHP

❖ PDO: consultas preparadas

1) Se prepara la consulta en el servidor con el método **prepare()** indicando, en este ejemplo, los parámetros mediante un **nombre precedido de dos puntos**.

2) Se asignan valores a los parámetros utilizando el método **bindParam()**. En el ejemplo a cada parámetro se le asigna una variable.

3) A continuación, en este ejemplo se asignan valores a las variables que se van a asignar a los parámetros.

4) Para cada juego de variables, se ejecuta la sentencia preparada.

```
try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // prepare sql and bind parameters
    1 $stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
    VALUES (:firstname, :lastname, :email)");
    2 $stmt->bindParam(':firstname', $firstname);
    $stmt->bindParam(':lastname', $lastname);
    $stmt->bindParam(':email', $email);

    // insert a row
    $firstname = "John";
    $lastname = "Doe";
    $email = "john@example.com";
    3 $stmt->execute();

    // insert another row
    $firstname = "Mary";
    $lastname = "Moe";
    $email = "mary@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Julie";
    $lastname = "Dooley";
    $email = "julie@example.com";
    $stmt->execute();

    echo "New records created successfully";
}
catch(PDOException $e)
{
    echo "Error: " . $e->getMessage();
}
$conn = null;
```